The International Journal of Engineering and Science (IJES)
|| Volume || 6 || Issue || 8 || Pages || PP 01-13 || 2017 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Development of a Mobile Agent System for Monitoring Memory Usage in a Network

Samson A. Arekete[1] Bosede O. Oguntunde[2] Oluwatosin G. Ore-Adewole[3]
Department of Computer Science, Redeemer's University, Ede, Nigeria
[1]areketes@run.edu.ng, [2]oguntunden@run.edu.ng, [3]grace2tee@gmail.com
Corresponding author: Samson A. Arekete

----------ABSTRACT----------

Memory management concerns the process of managing the computer memory, in particular, the main memory, which is an expensive resource of the computer system. In this study, a mobile agent was developed for monitoring memory usage in a network. Monitoring the memory usage of systems in a network is essential in a multiprocessing and multitasking environment due to fact that some processes invoked at startup or those running on the background can consume physical memory and reduce the available memory in their idle states; this way, they reduce the efficiency of the system. Monitoring memory helps to increase the performance of the system and enhance the efficiency of the network. The features of the mobile agent were modeled using the Unified Modelling Language. Programming was done in Java and implementation adopted the Java Agent Development Environment (JADE), a versatile mobile agent platform. A number of computer systems were paired and the monitoring agent system was successfully deployed.

Keywords:*Mobile agent,memory management, network, distributed systems, agent life cycle, monitoring.*

Date of Submission:15-07-2017 Date of Publication: 1-08-2017

I. INTRODUCTION

A modern computer system is a complex aggregation of one or more processors, some main memory, input/output devices (the standard being a keyboard and a display unit), disks, network interface and a number of peripheral devices such as printers, modem, scanner and mouse [1]. A computer system resource is any usable part of a computer that can be controlled and assigned by the operating system so that all of the hardware and software on the computer can work together as designed [2]. The resources of a computer system can be broadly classify into four[3]: the main memory, the central processing unit (CPU), the devices and the files. The operating system is in charge of managing all these resources. Memory management can be described as the process of managing the computer memory which consists of primary memory and secondary memory[4],[5]. It can be thought of as the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance[6]. Managing the main memory is the most critical since it is smaller than the secondary memory; the former is also costlier and faster. There are a number of memory management algorithms which take care of efficient allocation of the main memory as a scarce resource. It is also important to note that every application and process run by the computer system must go through the main memory. Memory management helps in the following ways:

a. It keeps track of which parts of memory are in use and which part is not in use.
b. It helps in the allocation and de-allocation of memory blocks to programs as requested by the user.
c. It ensures that there is at least enough memory for processes or programs to run.
d. It takes up the task of making sure that processes that require small size of memory blocks are allowed to take sole control of the memory before the ones that require large size of memory block.

Physical memory describes the size of random access memory (RAM) on a system. It contains the drivers and files. It is what the computer uses to load the operating system as well as processes, programs and application. For instance, a system can have a 1GB, 2GB, and 4GB of physical memory. Available physical memory is the amount or size of memory left after some programs have been invoked either by the user of the system or at startup.

Monitoring the memory helps to increase the performance of the system and enhances its efficiency. When several applications are opened on a system, each application uses part of the memory, thereby reducing the amount of available memory left. If the program or application is a large one, it consumes more memory than a smaller program and if several of such large applications are running, it reduces the efficiency of the system

making the system work slowly. It is essential therefore to monitor the memory in other to speed up the system and increase performance.The activities of a user can impact on the general network performance because some of the network resources are shared. For instance, a user who is downloading a heavy file will be consuming the bandwidth of the Internet subscribed to by the network since bandwidth is a shared resource. Of course, the download also has impact on the local memory of the user's system. When a node is running low on physical memory, the rate of download will also be slower and it would need more time to download a particular file size.

Mobile agent is an active object that can migrate from one node in a network to another and performing tasks in each node. Mobile agents possess some characteristics which include mobility, autonomous, scalability, etc. They reduce network traffic and bandwidth requirements, they migrate themselves according to the environment, they increase asynchrony between clients and servers etc. [7].A mobile agent can be used to monitor the memory usage of systems in a network. It can be launched on a node in the network to monitor the on-going activities on the entire network. The agent checks the available memory as well as the application running on it and evaluates it. If the available memory is too low to run another program, it checks for the processes that are not in use and those that are consuming much memory and end such programs or recommend to end them in order to free up some memory.It also gauges the health status of a node using the available memory. Monitoring the memory of a system in a network can be done manually but the use of a mobile agent is faster and more effective. Monitoring of memory usage helps to increase performance as well as efficiency and ensures the effectiveness of the network.

Resource and network monitoring have been investigated in[8]. The researchers noted that resource monitoring involves gathering information concerning system resource usage which are needed for improving the performance of the system; and described network monitoring as involving monitoring the bandwidth for better utilization of the resources including hard drive, RAM, and so on. That research proposed a mobile agent-based automated deployment of resource monitoring service in a grid system and took a novel approach for automated deployment of resource monitoring service for job submission in grid environment, which is to monitor manually because of the geographically distributed nodes. The mobile agent approach takes less time to deploy when compared to the manual deployment. The mobile agents do the automated deployment with minimum deployment time and bandwidth, and in turn reduces the network load.The advantages of using mobile agents include:

a. moving computation closer to the data, hence reducing network traffic;
b. reduction of the usage of network bandwidth; and
c. migration to network nodes for monitoring purposes.

In[9], the development of mobile agent for monitoring and evaluation of the activities of users in a network environment was presented. The mobile agent is migrated to nodes in the network to collect configuration and application data that reveals the pattern of users' activities in the network. Since the activities of a user on a node can affect the general performance of the entire network, mobile agent can help the network manager to identify users who consume too much bandwidth in downloads at the detriment of other users, for instance. These two studies motivated the current research.

The main purpose of this research is to develop a system that can effectively monitor the physical memory of a system and enhance its optimal performance of the network. Manual monitoring of network is cumbersome and ineffective in large networks. Monitoring based on client-server consumes much bandwidth and suffers from lack of flexibility. Therefore, mobile agent provides a good alternative, it can move around the network and monitor memory usage effectively.The aim of this study is to develop a mobile agent to monitor the physical memory usage of systems in a network environment.The specific objectives are:

a. To design a mobile agent system to monitor the usage of the physical memory of systems in the network.
b. To implement the mobile agent system using the JADE mobile agent platform
c. To carry out a case study of the Redeemer's University software Laboratory.

The following steps will be taken to achieve the aim and objectives:

a. An appropriate algorithm would be formulated to control the function of the mobile agent.
b. Modelling the behaviour and interaction of the mobile agent using UML.
c. Adopting the Java programming language to implement the mobile agent system, drawing in particular on its strength of object serialization for the migration of the agent.
d. JADE would be adopted as the mobile agent platform and NetBeans would be used as the integrated development environment (IDE).

The rest of the research is organized as follows. Section 2 presents the literature review of research work of other people related to mobile agents, network and memory monitoring. Section 3 details the design, research methodologies, and different approaches of solving the problem. In section 4 the implementation techniques of

the mobile agent system are presented, while section 5 presents the conclusion and directions for future research.

II. LITERATURE REVIEW

Monitoring is a network management process directed at ensuring optimal performance of the network. The term network monitoring describes the use of a system that constantly monitors a computer network to detect slow or failing components and that notifies the network administrator (via email, SMS and so on) in case of outages[5].

Mobile agents are computing software entities that can start operation on a network node, suspend action there and migrate to another node, and resume operations from where they left off on the previous node. A mobile agent is able to migrate autonomously to node. In concept, it is able to move its whole virtual machine from host to host; it owns its code but not the resources for execution which is owned by the host. While on a particular host, the mobile agent can carry out a number of tasks, and afterwards, it can migrate to another host to continue operation from where it stopped action on the previous host [9],[10].

The system memory is the place where the computer holds current programs and data that are in use. There are various levels of computer memory, including ROM, RAM, cache, page and graphics, each with specific objectives for system operation. In general, if there is a shortage in the physical memory and the paging file (swap file), which uses physical areas in RAM, the entire system's performance is affected adversely. However, a memory shortage is not the only bottleneck in the system. Monitoring memory usage helps you to detect network overloads at an early stage before they result in downtimes or data loss. It also helps to identify underused servers, and redistribute loads accordingly [11].

Java Agent DEvelopment Framework (JADE) is an agent development framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases [12]. "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors"[13]. According to Naylor (2000), an agent is defined as something which acts on behalf of somebody. It is assigned some goals to achieve and to do this, it may need to interact with other agents. Agent system is made up of an agent and its environment. Software agents have been studied widely. Agents display a number of both common and distinguishing characteristics that set them apart from ordinary software programs. One such characteristic, mobility, is envisaged to likely play an important role in future distributed systems[14].

Authors have classified agents into different categories – autonomous[15], intelligent[16], interface software[17], and mobile [9], [14]. In[18] a thorough and well thought out classification, in which they state that agents possess several or all of the following characteristics is provided: reactive, autonomous, goal-oriented, and temporally-continuous. Others include communicative, learning, mobile, and flexible. Every agent is expected to satisfy the first four properties (reactive, autonomous, goal-oriented and the temporal continuity characteristics).

According to [19] a mobile agent is defined as a software program that travels from one platform to another in order to get its work done, during this process, it carries its state and data with itself and resumes its execution from the state it had left off on the previous platform to the new platform. Existing mobile agent-based network management systems often assume that their mobile agents are designed to work in particular networks to raise the efficiency of agent migration among multiple nodes[20]. Adopting the mobile agent technology eliminates the need for the administrator to constantly manually monitor many network management activities. Such applications are required to migrate their agents among all specific nodes efficiently to perform their own tasks at each of the visited node [14]. Mobile agents can be useful in several areas and in particular, in the areas of e-commerce, banking applications, and network monitoring.

A number of the mobile agent models or frameworks are discussed in [21] and these include: Aglets, Concordia, Odyssey, Voyager and Grasshopper. JADE, the framework on which this research work is based, is a software platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction[22]. A significant merit of JADE is that it implements this abstraction over a well-known object-oriented language, Java, providing a simple and friendly API. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI (Graphical User Interface). The configuration can even be changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in Java language and the minimal system requirement is the version 1.4 of JAVA, the runtime environment or the JDK[23].

According to [12], The Foundation for Intelligent Physical Agents (FIPA) was established in 1996 as an international non-profit association to develop a collection of standards relating to software agent technology. At that time software agents were already very well known in the academic community but have to date received

only limited attention from commercial enterprises beyond an exploratory perspective. The consortium agreed to produce standards that would form the bedrock of a new industry by being usable across a vast number of applications.

FIPA specify the normative rules that allow a society of agents to inter-operate, that is, effectively exist, operate and be managed. It establishes the logical reference model for the creation, registration, location, communication, migration and operation of agents. The roles include:

1. **The Agent Management System (AMS):** The AMS is a mandatory component of an AP and is responsible for managing the operation of an AP, such as the creation and deletion of agents, and overseeing the migration of agents to and from the AP. Each agent must register with an AMS in order to obtain an AID which is then retained by the AMS as a directory of all agents present within the AP and their current state (e.g. active, suspended or waiting).
2. **Agent Platform (AP)**: This provides the physical infrastructure in which agents are deployed.
3. **Directory Facilitator (DF):** The DF is an optional component of an AP providing yellow-page services to other agents. It maintains an accurate, complete and timely list of agents and must provide the most current information about agents in its directory on a non-discriminatory basis to all authorized agents.
4. **Agent**: An agent is a computational process that inhabits an AP and typically offers one or more computational services that can be published as a service description. An agent must have at least one owner and must support at least one notion of identity which can be described using the FIPA Agent Identifier (AID) that labels an agent so that it may be distinguished unambiguously.
5. **Message Transport Service (MTS)**: The MTS is a service provided by an AP to transport FIPA ACL messages between agents on any given AP and between agents on different APs. Messages are provided with a transport envelope that comprises the set of parameters detailing, for example, to whom the message is to be sent.

III. SYSTEM ANALYSIS AND DESIGN

All the processes or programs running on a system make use of the physical memory space no matter how small the size is. The system administrator is interested in monitoring the memory usage on each node. He can login to a system that serves as a controller on which he launches a mobile agent that moves around the nodes in the network, gather information on the nodes returns to the admin console to deliver the information. The mobile agent migrates from one node to another with the view to monitoring usage of resources within the system thereby taking note of:

a. the total physical memory of the system,
b. the memory consumed by each process.
c. the total memory consumed by all the processes running on the node,
d. the size of the hard drive on the system, and
e. the available memory.

The information gathered from each node is then stored on a database and available to the system administrator. The Unified Modeling Language (UML) has been adopted as the modelling tool. UML is used to express the construct and relationships between complex systems [24],[25].

Classification of the health status of the system is based on the percentage of the available memory in line with [9] though with some modification is as depicted in Table 1.

Table 1 Health Status of the System

Percentage	Health Status
>= 80%	Perfectly Healthy
60-79%	Very Healthy
50-59%	Healthy
30-49%	Unhealthy
1-29%	Very Unhealthy

The proposed system generates statistics on the activities of each node on a corporate network which can help the system administrator to evaluate the level of memory usage of each node on the network to ensure optimal performance and also to evaluate the health status of the memory of the system. The system may thus serve as a veritable tool to enhance system efficiency and network reliability.

3.1 System Architecture

The architecture shows the relationship between the system controller which launches the mobile agent and how the agent migrates from one system to another within the network to gather information. The information gathered is later sent to the system controller.

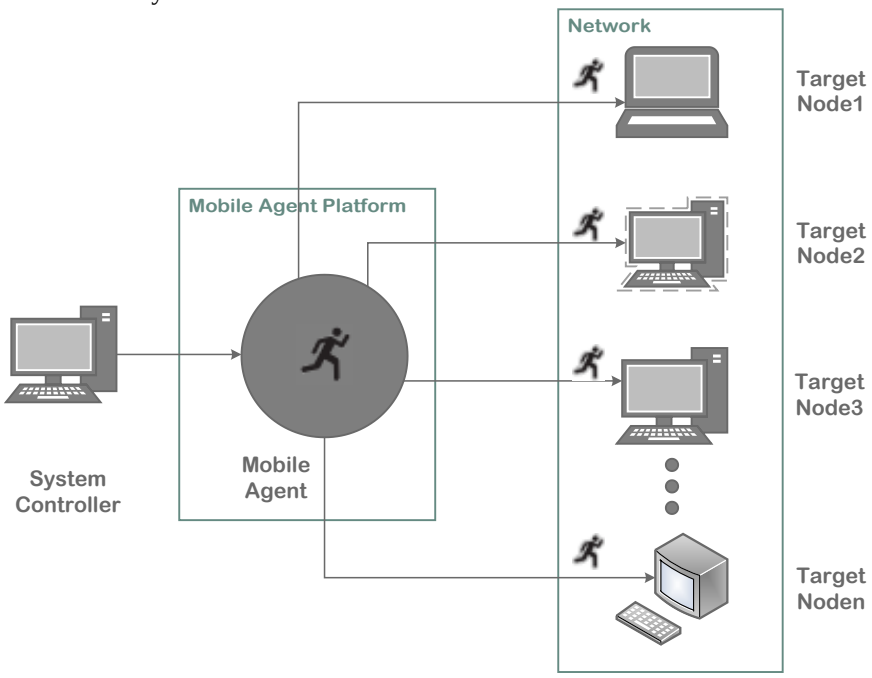

Figure 1 Architecture of Memory Monitoring System

3.2 Agent Life Cycle

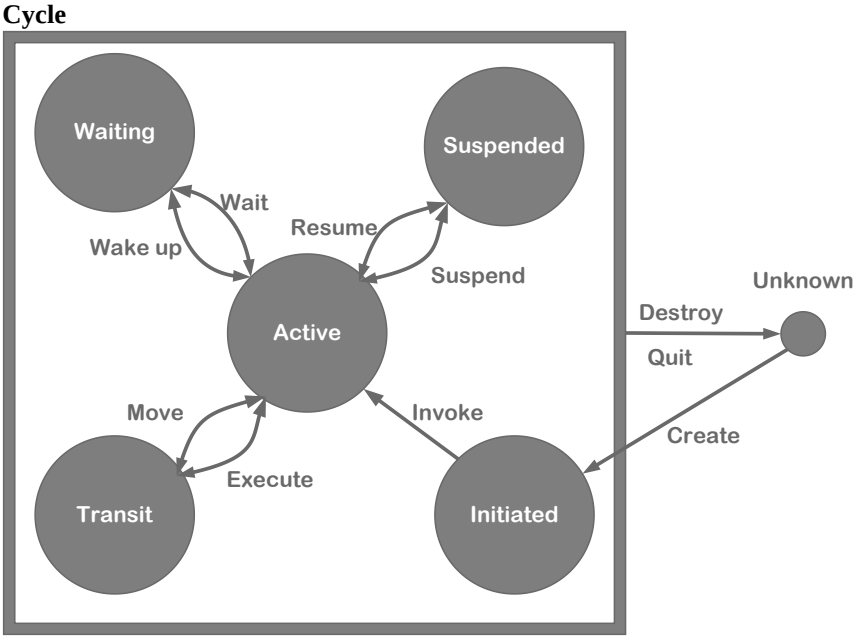

Figure 2 Mobile Agent Life Cycle (adapted from FIPA Agent Management, [26])

FIPA compliant agent can exist in one of several states, which include:

i. **INITIATED**: the agent object is built, but hasn't registered itself with AMS, has neither name nor an address and cannot communicate with other agent.
ii. **ACTIVE:** the agent object is registered with AMS, has regular name and address and can access all the various JADE features.

iii. **SUSPENDED:** the agent is currently stopped. Its internal thread is suspended and no agent behavior is being executed.
iv. **WAITING:** the agent is blocked waiting for something. Its internal thread is sleeping on a Java monitor and will wake up when some conditions are met (for example, when message arrives)
v. **DELETED:** the mobile agent is definitely dead. The internal thread has terminated its execution and the agent is no more registered with AMS.
vi. **TRANSIT:** the mobile agent enters this state while it is migrating to the new location. The system continues to buffer messages that will be sent to its new location

3.3 System Algorithm

The algorithm for the proposed system is as follows:

1. System Administrator logins into the controller system.
2. Mobile Agent is launched to the target node within the network.
3. Mobile Agent gathers information from the node and does this for all the nodes present in the network.
4. Mobile Agent returns to invoking controller system and returns the gathered information and displays on the screen.
5. Information is stored in the database.

3.4 Deployment Diagram

A deployment diagram is used to visualize the topology of the physical components of the system where software components are deployed. It shows how the hardware and software work together.

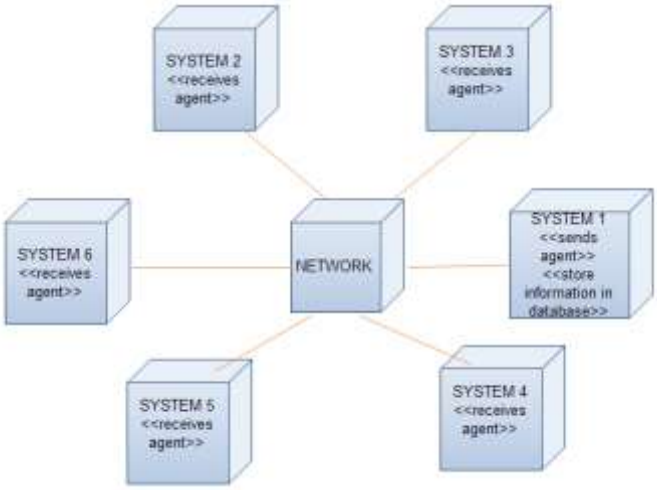

Figure 3 Deployment Diagram

The nodes exist in a network (Figure 3) and system 1 serves as the administrator's console with database access. System 1 launches the mobile agent to each of the systems and the information gathered migrates with the mobile agent to all the systems. The mobile agent returns back to system 1 with all the information and this is stored in the database.

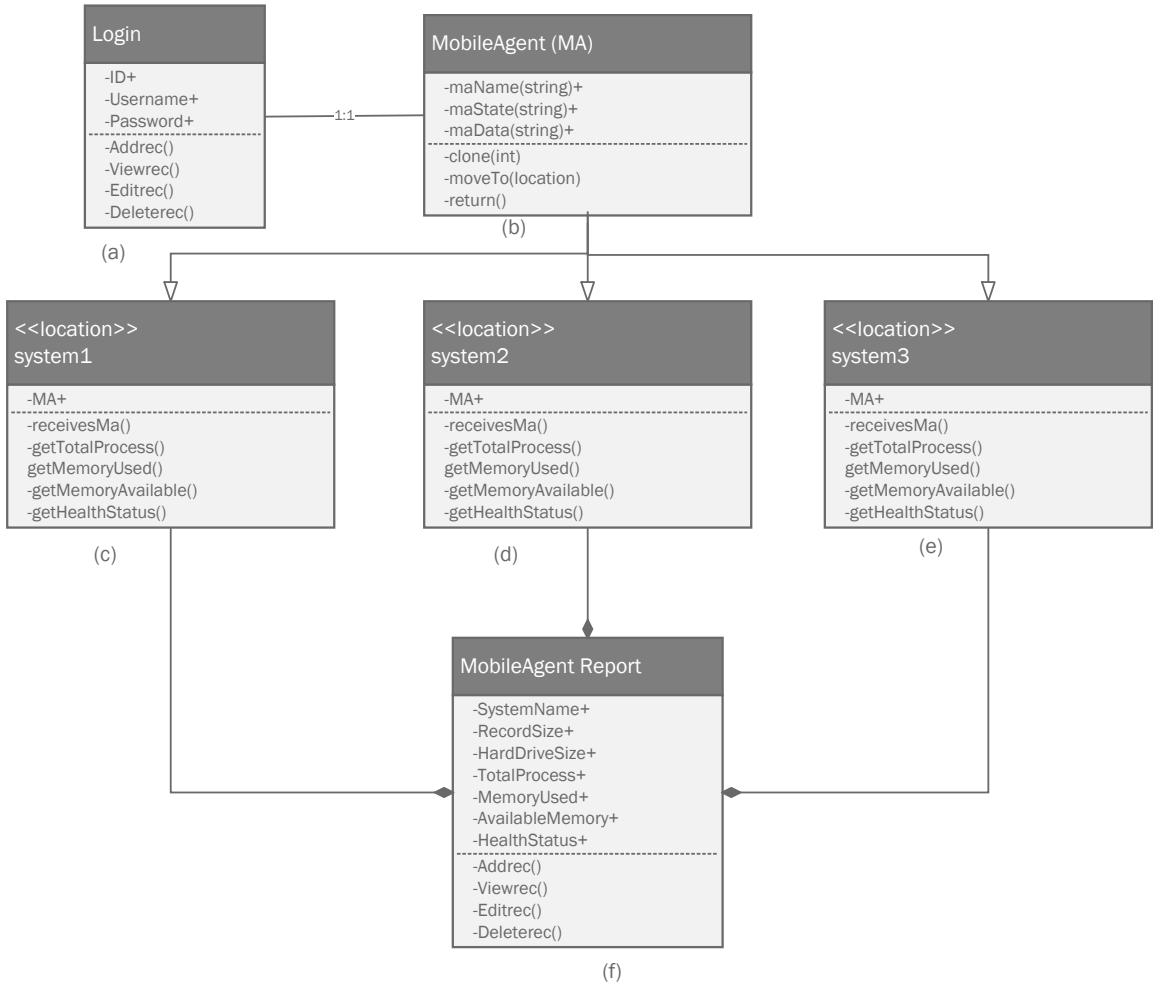

Figure 4 Class Diagrams

3.5 Class Diagrams

This diagram shows the static structure of classes and their possible relationships (i.e. association, inheritance, aggregation and dependency) in the system. It is adopted to model the static structural aspects of the design and process views of the system to model. The class diagram for the system is depicted in Figure 4.

The login class is shown in Figure 4a. The mobile agent class is shown in figure 4b while Figures 4c-e represents the cloned agent in specific nodes. Figure 4f depicts the agent report which is composed of data received from the different nodes.

3.6 Sequence Diagrams

This diagram describes the interactions of the various entities in the system and models the dynamic behavioural aspects of the user cases and actors in the system. The sequence diagram for this mobile agent system is depicted in Figure 5. From the sequence diagram, the mobile agent can visit one node and return with data or visit all the nodes and return with a comprehensive report.

In this diagram, the administrator launches or invokes the mobile agent, which then travels from one node to another. It gathers the information it gets from one node, migrates to another node with the information, to gather information of that particular new node. After visiting all the nodes, it then stores all the data gathered into the database at the controller system which can be accessed by administrator.

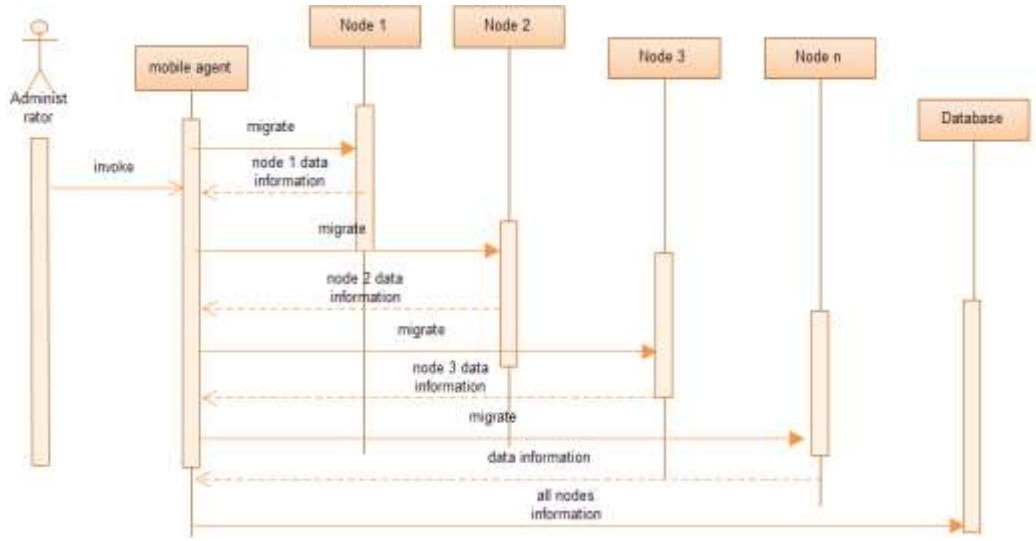

Figure 5 Sequence Diagram

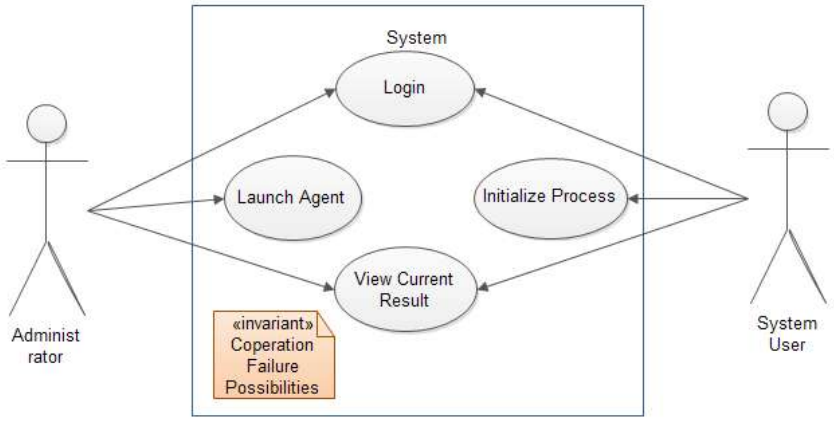

Figure 6 Administrator and System User's Use Cases

3.6 Use Case Diagrams

A use case diagram depicts a task an actor needs to perform with the help of a system. An actor is a user of a system that is playing a particular role. We identify two actors, the Administrator and the System User (Figure 6). The details of the use cases are shown in Tables 2 and 3.

Table 2 Administrator's Use Case Description

Use Case	Description
Login	The Administrator logins using username and password.
Launch Agent	The Administrator launches the memory monitoring agent.
View Result	The Administrator views the information gathered by the agent.

Table 3 System User's Description

Use Case	Description
Login	The System User logins into his/her own system using username and password
Initiate Process	Some processes are initiated at startup when a system user logins in and he can also initiate some processes himself
View Result	The System User can also view result.

3.7 Activity Diagram

The activity diagram for the mobile agent system is depicted in Figure 7. After a successful login, a mobile agent can be launched to move round the nodes to monitor the main memory. Results are deposited in the database after a roundtrip.

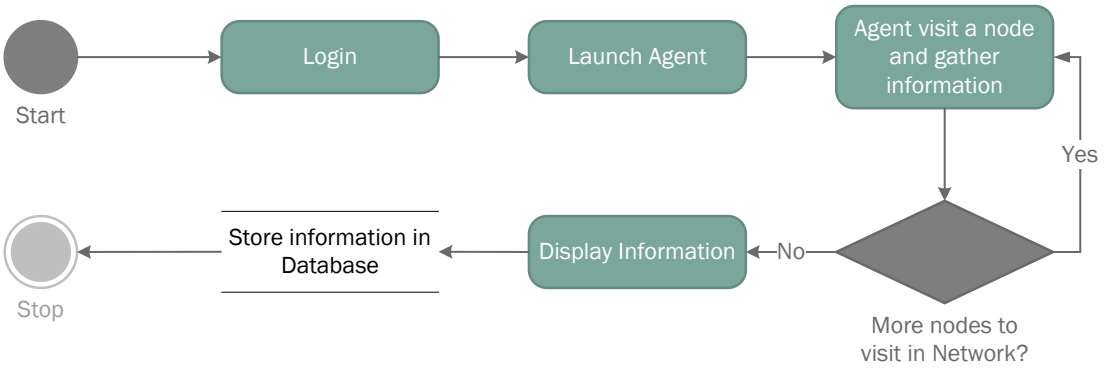

Figure 7 Activity Diagram of Mobile Agent

IV. SYSTEM IMPLEMENTATION

The mobile agent for monitoring memory can run on any system with the following minimum configuration: Intel Pentium IV CPU, 1 GB random access memory and a network interface. The software environment should include Java runtime environment, JADE platform and Windows XP operating system or higher.Access to the system is granted through the login form depicted in Figure 8. When the login is successful, the main menu depicted in Figure 9 is displayed. We can then decide to click a button to either dispatch an agent to the network, view the database or quit the system.

Figure 8 Admin Login Form

Figure 9 Main Menu Form

In the following, we present a scenario that depicts the operation of the mobile agent. The sendAgent is created on the console and sent to the nodes to be monitored (Figure 10).

Figure 10SentAgent Name and Class

The agent is added to the JADE Main-Container in Figure 11.

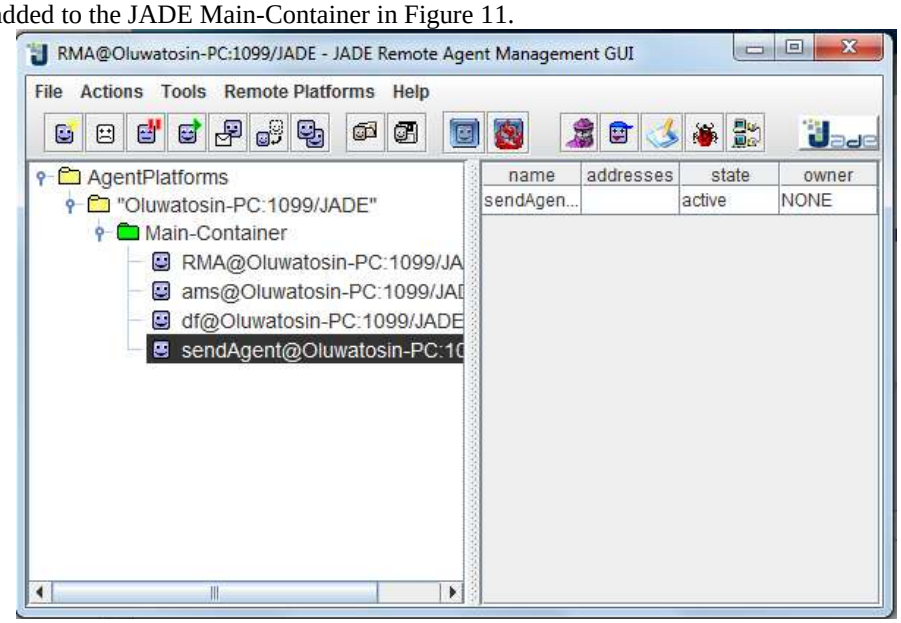

Figure 11sendAgent in the Main-Container

When launched, the Main-Container dispatches the sendAgent to the network (Figure 12). The sendAgent can collect information across the network nodes and deliver them to the database.

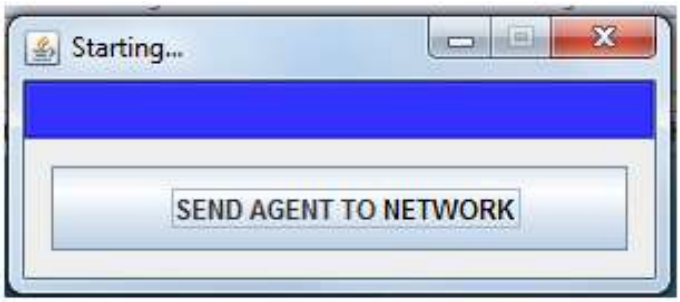

Figure 12sendAgent

On the alternative, the administrator can also obtain the information obtained from the monitored nodes by the sendAgent using the getAgent depicted in Figure 13. The agent name and class is supplied and added to the Main-Container of the administrator (Figure 14).

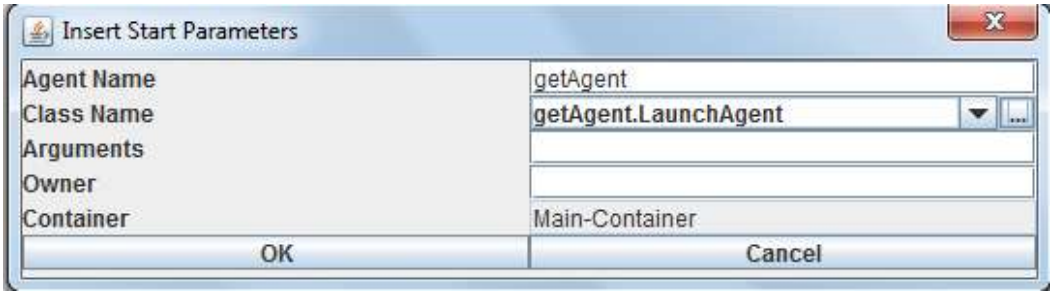

Figure 13getAgent name and class

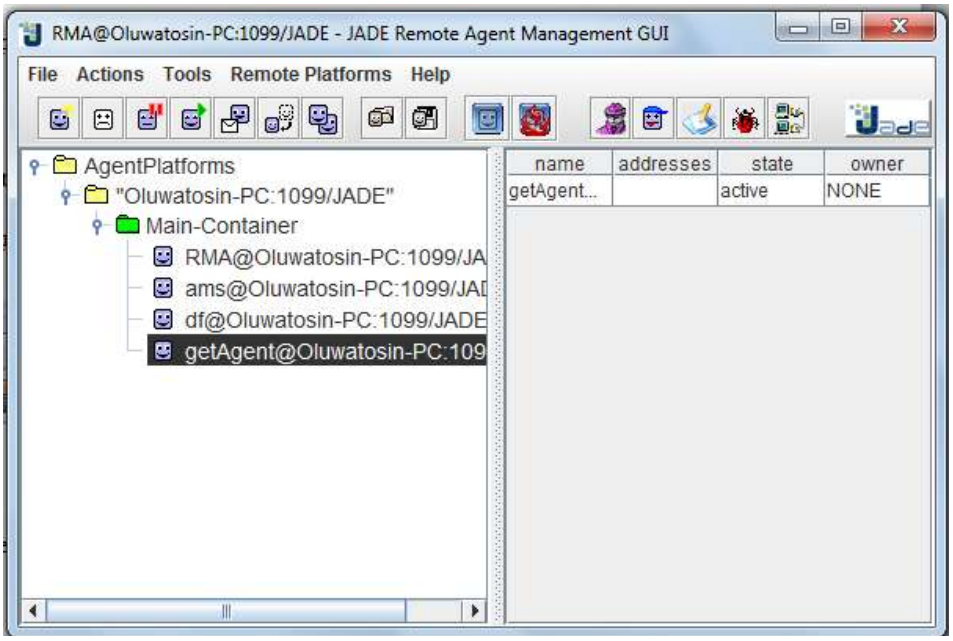

Figure 14getAgent in the Main-Container

Figure 15 depicts the launching of the getAgent in the System Console. The data obtained from the visited nodes are displayed in Figures 16 to 17.

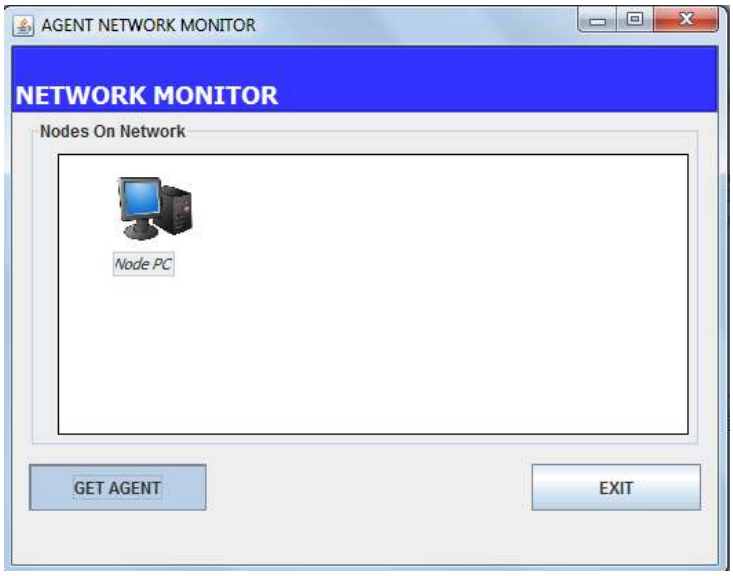

Figure 15getAgent on the System Console

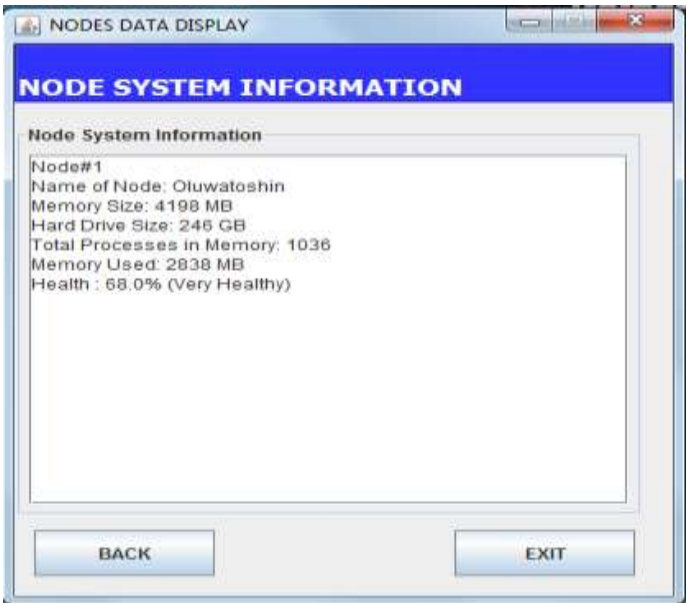

Figure 16 Information from Node 1

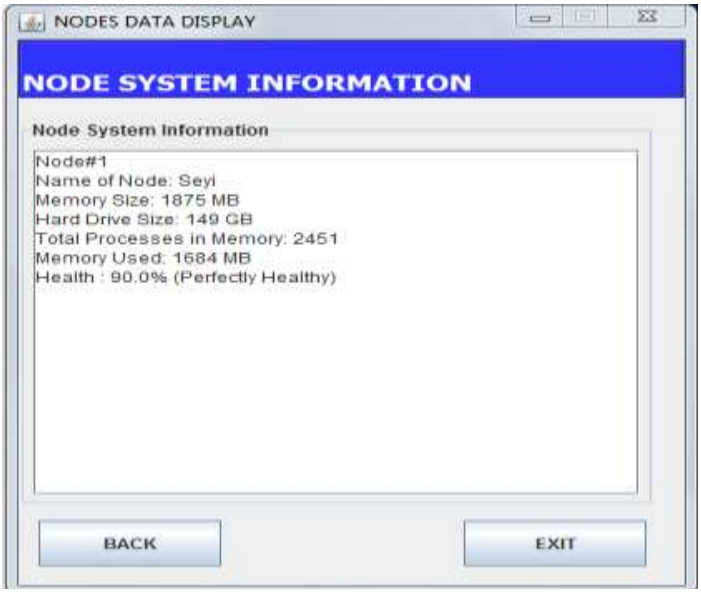

Figure 17 Information from Node 2

V. CONCLUSION AND FUTURE WORKS

This paper described the development of a mobile agent application tested using the local area network of software lab of the Redeemer's University. The system was modelled with the UML tools and implementation was made using the Java programming language and the Java Agent Development toolkit. The mobile agent developed can audit a network to monitor the systems' memory usage. The parameters of interest are the size of the physical memory (RAM), the size of the hard drive installed on the system, the total number of processes or applications running on the system, the total memory consumed, the available memory and the health status of the system. The system may serve as a tool to enhance network efficiency and reliability.

The software can be made to be more robust by incorporating the capabilities not only monitor to the memory of the system but to also terminate some applications that occupy memory and bug down the system. The current functionalities allow the user to take the initiative of terminating offending programs if the mobile agent indicates that the health status is impaired. Those improvements can be pursued in future research.

REFERENCES

[1] A. S. Tanenbaum, *Modern Operating System*, 2nd ed. USA: Prentice-Hall, 2001.
[2] T. Fisher. (2013, March 24). *Computer System Resources*. Available: www.about.com
[3] S. A. Arekete, *A Handbook on NRC Tower*. Akure, Nigeria: Hope Printers, 1996.
[4] S. Gibson. (1988, August) Tech Talk: Placing the IBM/Microsoft XMS Spec Into Perspective. *InfoWorld*.
[5] Wikipedia. (2017, March 24). *Memory management*. Available: www.wikipedia.com
[6] Gibillisco, "Memory Management," *Tech Taget*, Accessed on: March 24 2017Available: www.techtaget.com
[7] C. Valliyammai and S. S. Thamarai, "Mobile Agent-based Automated Deployment of Resource Monitoring Service in Grid," *Ubiquitous Computing and Communication Journal*, vol. 6, no. 2, pp. 786-790, 2011 2011.
[8] C. Valliyammai, S. Thamarai Selvi, S. Aravindh, J. Jervin, and K. Karthick, "Efficient Management of Network Resources Using Autonomous Agents in Grid," *International Journal of Computer Science and Network Security*, vol. 8, no. 4, pp. 178-184, April 2008 2008.
[9] S. A. Arekete, O. C. Akinyokun, O. Olabode, and B. K. Alese, "Design of a mobile agent for monitoring activities of users," *Journal of Computer Engineering and Intelligent Systems*, vol. 4, no. 2, pp. 33-48, 2013.
[10] A. Aneiba and S. J. Reels, "Mobile Agent Technology and Mobility," presented at the Proceedings of the 5th annual postgraduate symposium of the convergence of telecommunication, networking and broadcasting, Faculty of Computing, Engineering and Technology, Staffordshire University, Beaconside, Stafford ST18 ODG, UK, 2004.
[11] E. Bott. (2013, March 20, 2013). *System Memory*. Available: www.znet.com
[12] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. West Sussex PO19 8SQ, England: Wiley Series in Agent Technology, 2007.
[13] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. NJ. Prentice Hall: Eaglewood Cliffs, 1995.
[14] M. Naylor, "The Use of Mobile Agent in Network Management Application," M.Sc Thesis M.Sc, Information Technology (Systems Integration) School of Computing, Napier University, Edinburg, Scotland, 2000.
[15] C. Bohoris, "Network Performance Management Using Mobile Agent," PhD Thesis, Centre for Communication System Research School of Electronic and Physical Sciences, University of Surrey, Surrey, UK, 2003.
[16] IBM, "IBM Aglets Software Development Kit," USA1998, Available: http://www.trl.ibm.co.jp/aglets/.
[17] M. Watson, *Intelligent Java Applications for the Internet and Intranets*. Morgan Kauffman Publishers, 1997.
[18] S. Franklin and A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," in *Third International Workshop on Agent Theories, Architectures, and Languages*, 1996, pp. 21-35: Springer-Verlag.
[19] D. B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents," *Communications of the ACM*, vol. 42, no. 3, pp. 88-89, 1999.
[20] I. Satoh, "Building reusable mobile agents for network management " *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* vol. 33, no. 3, pp. 1-10, 2003.
[21] A. Karmouch and V. A. Pham, "Mobile agents: An Overview " *IEEE Communications Magazine*, vol. 33, no. 7, 1998 1998.
[22] M. J. Wooldridge and N. R. Jennings, "Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995.
[23] JADE. (2008, January 1, 2013). *JAVA Agent DEvelopment Framework open source platform for peer-to-peer agent based applications* Available: http://jade.tilab.com
[24] B. P. Douglas, P. D. Harel, Ed. *Real Time UML: Advances in the UML for Real-Time Systems*, Third ed. Boston, MA, USA: Addison Wesley, 2004.
[25] P. Stevens and R. Pooley, *Using UML. Software Engineering with Objects and Components*, 2nd ed. Edinburgh Gate, Harlow, England: Pearson Education Limited, 2006.
[26] FIPA. (2001, March 5). *FIPA Agent Management Specification*. Available: http://www.fipa.org/specs/fipa00023/

Samson A. Arekete. "Development of a Mobile Agent System for Monitoring Memory Usage in a Network." The International Journal of Engineering and Science (IJES) 6.8 (2017): 01-13.